Dirk Praetorius
Christoph Überhuber

Wintersemester 2006/07
27. Oktober 2006

Übungen zur Vorlesung
Einführung in das Programmieren für TM

Serie 3

Die Aufgaben mit Stern (*) sind bis zur nächsten Übung vorzubereiten und werden dort abgeprüft. Die übrigen Aufgaben dienen nur Ihrer Übung und mir als zusätzliche Grundlage für den Prüfungsstoff in den schriftlichen Tests. Kopieren Sie bitte den Source-Code in ein Unterverrzeichnis `serie03` *Ihres Home-Verzeichnisses. Überprüfen Sie bitte vor der Übung, ob Ihre Source-Codes mit dem* `gcc` *compiliert werden können. In den folgenden Aufgaben 17-26 sollen im wesentlichen* ***Zählschleifen*** *geübt werden. Außer Verzweigungen und elementarer Arithmetik sind keine weiteren Elemente von C nötig. Aufgabe 27 übt noch einmal elementare Verzweigungen sowie das Einbinden der mathematischen Bibliothek. Aufgabe 28 ist eine erste Aufgabe zu Pointern.*

Aufgabe 17*. Man schreibe eine nicht-rekursive Funktion `fibonacci`, die zu gegebenem Index k das Folgenglied x_k zurückgibt. Ferner schreibe man ein aufrufendes Hauptprogramm, das den Index n einliest und x_n ausgibt. Dabei soll die Fibonnaci-Folge *nicht* als Vektor gespeichert werden. Den Source-Code speichere man unter `fibonacci.c` ins Verzeichnis `serie03`. Ist diese Implementierung effizienter als die rekursive Implementierung aus Aufgabe 13?

Aufgabe 18*. Man schreibe eine nicht-rekursive Funktion `binomial`, die den Binomialkoeffizienten $\binom{n}{k}$ berechnet. Dazu realisiere man die gekürzte Form $\binom{n}{k} = \frac{n\cdot(n-1)\cdots(n-k+1)}{k\cdot(k-1)\cdots 1}$ mittels geeigneter Schleifen. Man schreibe ein aufrufendes Hauptprogramm, in dem $k, n \in \mathbb{N}_0$ mit $k \leq n$ eingelesen werden und $\binom{n}{k}$ ausgeben wird. Den Source-Code speichere man unter `binomial.c` ins Verzeichnis `serie03`. Welche der drei Implementierungen (vgl. Aufgabe 11) führt auf die effizienteste Variante und warum?

Aufgabe 19*. Man schreibe eine Funktion `skalarprodukt`, die zu gegebenen Vektoren $x, y \in \mathbb{R}^n$ das Skalarprodukt $x \cdot y := \sum_{j=1}^{n} x_j y_j$ berechnet. Ferner schreibe man ein aufrufendes Hauptprogramm, das die Vektoren x und y einliest und $x \cdot y$ ausgibt. Die Länge n der Vektoren soll eine Konstante im Hauptprogramm sein, die Funktion `skalarprodukt` ist für beliebige Länge n zu programmieren. Speichern Sie den Source-Code unter `skalarprodukt.c` ins Verzeichnis `serie03`.

Aufgabe 20*. Man schreibe eine Funktion `max`, die von einem gegebenem Vektor $x \in \mathbb{R}^n$ das Maximum $\max_{j=1}^{n} x_j$ berechnet und zurückgibt. Ferner schreibe man ein aufrufendes Hauptprogramm, das den Vektor x einliest und das Maximum ausgibt. Die Länge des Vektors soll eine Konstante im Hauptprogramm sein, die Funktion `max` ist für beliebige Länge n zu programmieren. Den Source-Code speichere man unter `max.c` ins Verzeichnis `serie03`.

Aufgabe 21*. Ein Tripel $(x, y, z) \in \mathbb{N}^3$ natürlicher Zahlen heißt *pythagoräisches Zahlentripel*, falls $x^2 + y^2 = z^2$ gilt. Das wohl bekannteste Beispiel ist $(3, 4, 5)$. Offensichtlich gelten $z > \max\{x, y\}$ sowie $x \neq y$ und ohne Beschränkung der Allgemeinheit ferner $x < y$. Man schreibe eine Prozedur `pythagoras`, die zu gegebener Schranke $n \in \mathbb{N}$ alle pythagoräischen Zahlentripel mit $x < y < z \leq n$ bestimmt und ausgibt. Ferner schreibe man ein aufrufendes Hauptprogramm, in dem die Schranke n eingelesen wird. Den Source-Code speichere man unter `pythagoras.c` ins Verzeichnis `serie03`.

Aufgabe 22. Man schreibe eine Funktion `prim`, die überprüft ob eine natürliche Zahl $n \in \mathbb{N}$ eine Primzahl ist (Rückgabewert 1) oder nicht (Rückgabewert 0). Ferner schreibe man ein aufrufendes Hauptprogramm, das den Wert n von der Tastatur einliest und am Bildschirm ausgibt, ob n eine Primzahl ist.

Aufgabe 23. Man schreibe eine Funktion `mean`, die den Mittelwert $\frac{1}{n}\sum_{j=1}^{n} x_j$ eines Vektors $x \in \mathbb{R}^n$ zurückgibt. Ferner schreibe man ein aufrufendes Hauptprogramm, das den Vektor x einliest und den Mittelwert ausgibt. Die Länge n des Vektors soll eine Konstante im Hauptprogramm sein, die Funktion `mean` ist für beliebige Länge zu programmieren.

Aufgabe 24. Um die Summe $\sum_{j=1}^{n}(-1)^j/j$ zu berechnen, ist es numerisch günstig, zunächst die negativen und die positiven Beiträge getrennt zu summieren und erst abschließend beide Teilsummen zu addieren. Warum? Man schreibe eine Funktion `sum`, die dieses Vorgehen realisiert. Ferner schreibe man ein Hauptprogramm, das n einliest und $\sum_{j=1}^{n}(-1)^j/j$ ausgibt.

Aufgabe 25. Die Gleitpunkt-Arithmetik ist nicht assoziativ. Das numerische Ergebnis der Summe $\sum_{j=0}^{n} x^j/j!$ hängt daher von der Summationsreihenfolge ab. Man schreibe Funktionen `forward` und `backward`, die diese Summe auf zwei Weisen berechnen: `forward` entspricht Vorwärtssummation $j = 1, \ldots, n$, `backward` entspricht Rückwärtssummation $j = n, \ldots, 1$. Welche Variante wird besser sein – zumindest für positives x und großes n?

Aufgabe 26. *Bubble-Sort* ist ein ineffizienter, aber kurzer Sortier-Algorithmus: Man vergleicht aufsteigend jedes Element eines Arrays x_j mit seinem Nachfolger x_{j+1} und - falls notwendig - vertauscht die beiden. Nach dem ersten Durchlauf muß zumindest das letzte Element bereits am richtigen Platz sein. Der nächste Durchlauf muß also nur noch bis zur vorletzten Stelle gehen, usw. Wie viele geschachtelte Schleifen braucht dieses Vorgehen? Man schreibe eine Funktion `x = bubblesort(x)`, die ein gegebenes Array $x \in \mathbb{R}^n$ mittels Bubble-Sort aufsteigend sortiert, d.h. $x_1 \leq x_2 \leq \cdots \leq x_n$, und zurückgibt. Ferner schreibe man ein aufrufendes Hauptprogramm, das den Vektor x einliest und in sortierter Reihenfolge ausgibt. Die Länge des Vektors soll eine Konstante im Hauptprogramm sein, die Funktion `bubblesort` ist für beliebige Länge n zu programmieren. Was ist der Vorteil von Bubble-Sort gegenüber Selection-Sort aus der Vorlesung?

Aufgabe 27. Man schreibe eine Prozedur `kurvendiskussion`, die für eine Parabel $p(x) = a_0 + a_1 x + a_2 x^2$ mit Koeffizientenvektor $a \in \mathbb{R}^3$ eine Kurvendiskussion durchführt. Wenn vorhanden, berechne man das Extremum (und Art) und die Nullstellen. Anderenfalls gebe man aus, dass die Parabel kein Extremum bzw. keine Nullstelle besitzt. Man schreibe ferner ein aufrufendes Hauptprogramm, das den Vektor a einliest und die Prozedur aufruft.

Aufgabe 28. Man schreibe eine Funktion `minmax`, die von einem gegebenem Vektor $x \in \mathbb{R}^n$ das Minimum $\min_{j=1}^{n} x_j$ und das Maximum $\max_{j=1}^{n} x_j$ berechnet und zurückgibt. Ferner schreibe man ein aufrufendes Hauptprogramm, das den Vektor x einliest und das Maximum ausgibt. Da eine C-Funktion maximal einen elementaren Datentyp als Return-Wert zurückliefern kann, realisiere man die Rückgabe mittels Call by Reference (Pointer!). Die Länge n des Vektors soll eine Konstante im Hauptprogramm sein, die Funktion `minmax` ist für beliebige Länge zu programmieren.